Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (currently amended): A method in a data processing system including a software development project having a plurality of elements, each element having corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including a first element and second element by using the names and display coordinates correlated by the transient meta model;

receiving a request to form a link;

receiving an indication of a first of the plurality of elements;

receiving an indication of a second of the plurality of elements; and

in response to receiving the request, the indication of the first element, and the indication of the second element,

generating new code independent of the graphical representation and

adding the new code to the first element to reflect the link to the

second element and

modifying the graphical representation of the code associated with

the first element to reflect the link to the second element.

2. (cancelled).

3. (cancelled).

4. (original): The method of claim 1, wherein the step of adding new code to the first element comprises the steps of:

determining whether linking the first element to the second element would violate

a predefined rule; and

when it is determined that linking the first element to the second element would not violate a predefined rule,

adding the new code to the first element to form the link to the second

element.

5. (original): The method of claim 4, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is a class and whether the second element is

another class; and

when it is determined that the first element is the class and that the second element is the other class,

identifying the link from the first element to the second element as an inheritance link.

6. (original): The method of claim 5, further comprising the step of identifying a link error when it is determined that the first element is the class and that the second element is not the other class.

7. (original): The method of claim 4, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is a class and whether the second element is an interface; and

when it is determined that the first element is the class and that the second element is the interface,

identifying the link from the first element to the second element as an implementation link.

8. (original): The method of claim 7, further comprising the step of identifying a link error when it is determined that the first element is the class and that the second element is not the interface.

9. (original): The method of claim 4, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is an interface and the second element is another interface; and

when it is determined that the first element is the interface and the second element is the other interface,

identifying the link from the first element to the second element as an inheritance link.

10. (original): The method of claim 9, further comprising the step of identifying a link error when it is determined that the first element is the interface and the second element is not the other interface.

11. (cancelled).

12. (cancelled).

13. (cancelled).

14. (cancelled).

15. (cancelled).

16. (cancelled).

17. (cancelled).

18. (cancelled).

19. (cancelled).

20. (currently amended): A method in a data processing system including a software development program having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names along with a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the source and the destination by using

the names and display coordinates correlated by the transient meta model;

receiving an identification of the link;

receiving a selection of one of the linked elements;

receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

determining whether the selected element is the destination; and

when it is determined that the selected element is the destination,

modifying the corresponding code of the other element independently of the graphical representation in order to reflect a new

link between the other element and the destination element and

modifying the graphical representation of the corresponding code of the other element to reflect the new link between the other element and the destination element.

21. (original): The method of claim 20, wherein the modifying step further includes the step of modifying the corresponding code of the source to reflect the removal of the link between the source and the destination.

22. (cancelled).

23. (original):The method of claim 184, further comprising the step of modifying the graphical representation of the corresponding code of the source to reflect the removal of the link between the source and the destination.

24. (cancelled).

25. (cancelled).

26. (original): The method of claim 20, wherein the modifying step includes the steps of:

determining whether linking the other element to the destination would violate a predefined rule; and

when it is determined that linking the other element to the destination would not violate a predefined rule,

modifying the corresponding code of the source to reflect the removal of the link between the source and the destination; and

adding new code to the corresponding code of the other element to reflect the new link between the other element and the destination element.

27. (original): The method of claim 26, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is a class and whether the destination is

another class; and

when it is determined that the other element is the class and that the destination is the other class,

identifying the new link between the other element and the destination as an inheritance link.

28. (original): The method of claim 26, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is a class and whether the destination is an interface; and

when it is determined that the other element is the class and that the destination is the interface,

identifying the new link between the other element and the destination as an implementation link.

29. (original): The method of claim 28, further comprising the step of identifying a link error when it is determined that the other element is the class and that the destination is not the interface.

30. (original): The method of claim 26, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is an interface and the destination is another interface; and

when it is determined that the other element is the interface and the destination is the other interface,

identifying the new link between the other element and the destination as an inheritance link.

31. (original): The method of claim 30, further comprising the step of identifying a link error when it is determined that the other element is not the interface.

32. (original): The method of claim 30, further comprising the step of identifying a link error when it is determined that the destination is not the other interface.

33. (cancelled).

34. (cancelled).

35. (cancelled).

36. (cancelled).

37. (cancelled).

38. (cancelled).

39. (cancelled).

40. (cancelled).

41. (cancelled).

42. (currently amended): A method in a data processing system including a software development project having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names along with a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the source and destination by using the names and display coordinates correlated by the transient meta model;

receiving an identification of the link;

receiving a selection of one of the linked elements;

receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

determining whether the selected element is the source; and

when it is determined that the selected element is the source,

modifying the corresponding code of the source independently of the graphical representation in order to reflect a new link between the source and the other element and

modifying the graphical representation of the corresponding code of the source element to reflect the new link between the source element and the other element.

43. (cancelled).

44. (previously presented): The method of claim 184, further comprising the step of modifying the graphical representation of the code corresponding the source to reflect the removal of the link to the destination.

45. (original): The method of claim 42, further comprising the steps of:

when it is determined that the selected element is the source,

determining whether linking the source to the other element would violate a predefined rule; and

when it is determined that linking the source to the other element would not violate a predefined rule,

modifying the corresponding code of the source to reflect the removal of the link between the source and the destination; and

adding new code to the corresponding code of the source to reflect the new link to the other element.

46. (cancelled).

47. (original): The method of claim 45, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is a class and whether the other element is another class; and

when it is determined that the source is the class and that the other element is the other class,

identifying the new link between the source and the other element as an inheritance link.

48. (original): The method of claim 45, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is a class and whether the other element is an interface; and

when it is determined that the source is the class and that the other element is the interface,

identifying the new link from the source to the other element as an implementation link.

49. (original): The method of claim 48, further comprising the step of identifying a link error when it is determined that the other element is not the interface.

50. (original): The method of claim 45, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is an interface and the other element is another interface; and

when it is determined that the source is the interface and the other element is the other interface,

identifying the new link between the source and the other element as an inheritance link.

51. (original): The method of claim 50, further comprising the step of identifying a link error when it is determined that the source is not the interface.

52. (original): The method of claim 50, further comprising the step of identifying a link error when it is determined that the other element is not the other interface.

53. (cancelled).

54. (cancelled).

55. (cancelled).

56. (cancelled).

57. (cancelled).

58. (cancelled).

59. (cancelled).

60. (cancelled).

61. (cancelled).

62. (cancelled).

63. (cancelled).

64. (cancelled)..

65. (cancelled).

66. (cancelled).

67. (cancelled).

68. (currently amended): A method in a data processing system including a software development project having a plurality of elements having a graphical representation stored in a graphical view file that includes symbols having display coordinates and a corresponding code stored in a source code file that includes names, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including a first and second element by using the names and display coordinates correlated by the transient meta model;

receiving an identification of a first of the plurality of elements;

receiving an identification of a second of the plurality of elements;

receiving an indication that the first element is to be included in the second element;

determining whether the inclusion of the first element in the second element would violate a predefined rule; and

when it is determined that the inclusion of the first element in the second element would not violate a predefined rule,

transferring the code corresponding to the first element into the second element, wherein said code transfer occurs independently of the graphical representation and

modifying the graphical representation of the code of the second element to reflect the transfer of the code corresponding to the first element into the second element.

69. (cancelled).

70. (cancelled).

71. (previously presented): The method of claim 68, wherein the step of transferring code comprises the steps of:

removing the code corresponding to the first element from a file;

placing the code corresponding to the first element within the code corresponding

to the second element; and

deleting the file corresponding to the first element.

72. (previously presented) The method of claim 68, wherein the method further comprises the steps of:

when it is determined that the first element is the class and that the second element is not another class,

determining whether the second element is a package; and

when it is determined that the second element is a package,

moving a file that includes code corresponding to the first element to a directory associated with the second element.

73. (cancelled).

74. (cancelled).

75. (currently amended): A method in a data processing system including a software development project having a plurality of elements, each element having corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates, wherein code corresponding to a first of the plurality of elements is nested in the code corresponding to a second of the plurality of elements, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the first element and the second element by using the names and display coordinates correlated by the transient meta model;

receiving an indication that the first element is to be removed from the second element;

determining whether the removal of the first element from the second element would not violate a predefined rule; and

when it is determined that the removal of the first element from the second element would not violate a predefined rule,

removing code corresponding to the first element from the second element, wherein said code removal occurs independently of the graphical representation and

modifying the graphical representation of the code corresponding to the second element to reflect the removal of the first element from the second element.

76. The method of claim 75, further comprising the step of placing the code corresponding to the first element into a file.

77. (cancelled).

78. (cancelled).

79. (previously presented): The method of claim 75, further comprising the steps of:

when it is determined that the first element is a class and that the second element is not another class,

determining whether the second element is a package; and

when it is determined that the second element is a package,

removing a first file that includes code corresponding to the first element from a directory associated with the second element and placing the first file in another directory.

80. (cancelled).

81. (cancelled).

82. (cancelled).

83. (currently amended): A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system including a software development project having a plurality of elements, each element having corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the first element and the second element by using the names and display coordinates correlated by the transient meta model;

receiving a request to form a link;

receiving an indication of a first of the plurality of elements;

receiving an indication of a second of the plurality of elements; and

in response to receiving the request, the indication of the first element, and the indication of the second element,

generating new code independent of the graphical representation and adding the new code to the first element to reflect the link to the second element and

modifying the graphical representation of the code associated with the first element to reflect the link to the second element.

84. (cancelled).

85. (cancelled).

86. (original): The computer-readable medium of claim 83, wherein the step of adding new code to the first element comprises the steps of:

determining whether linking the first element to the second element would violate a predefined rule; and

when it is determined that linking the first element to the second element would not violate a predefined rule,

adding the new code to the first element to form the link to the second element.

87. (original): The computer-readable medium of claim 86, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is a class and whether the second element is another class; and

when it is determined that the first element is the class and that the second element is the other class,

identifying the link from the first element to the second element as an inheritance link.

88. (original): The computer-readable medium of claim 87, wherein the method further comprises the step of identifying a link error when it is determined that the first element is the class and that the second element is not the other class.

89. (original): The computer-readable medium of claim 86, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is a class and whether the second element is an interface; and

when it is determined that the first element is the class and that the second element is the interface,

identifying the link from the first element to the second element as an implementation link.

90. (original): The computer-readable medium of claim 89, wherein the method further comprises the step of identifying a link error when it is determined that the first element is the class and that the second element is not the interface.

91. (original): The computer-readable medium of claim 86, wherein the step of determining whether linking the first element to the second element would violate a predefined rule comprises the steps of:

determining whether the first element is an interface and the second element is another interface; and

when it is determined that the first element is the interface and the second element is the other interface,

identifying the link from the first element to the second element as an inheritance link.

92. (original): The computer-readable medium of claim 91, wherein the method further comprises the step of identifying a link error when it is determined that the first element is the interface and the second element is not the other interface.

93. (cancelled).

94. (cancelled).

95. (cancelled).

96. (cancelled).

97. (cancelled).

98. (cancelled).

99. (cancelled).

100. (cancelled).

101. (cancelled).

102. (currently amended): A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system including a software development project having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names along with a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the source and destination by using the

names and display coordinates correlated by the transient meta model;

receiving an identification of the link;

receiving a selection of one of the linked elements;

receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

determining whether the selected element is the destination; and

when it is determined that the selected element is the destination,

modifying the corresponding code of the other element independently of the graphical representation in order to reflect a new link between the other element and the destination element and

modifying the graphical representation of the corresponding code of the other element to reflect the new link between the other element and the destination element.

103. (original): The computer-readable medium of claim 102, wherein the modifying step further includes the step of modifying the corresponding code of the source to reflect the removal of the link between the source and the destination.

104. (cancelled).

105. (previously presented): The computer-readable medium of claim 102, wherein the method further comprises the step of modifying the graphical representation

of the corresponding code of the source to reflect the removal of the link between the source and the destination.

106. (cancelled).

107. (cancelled).

108. (original): The computer-readable medium of claim 102, wherein the modifying step includes the steps of:

determining whether linking the other element to the destination would violate a predefined rule; and

when it is determined that linking the other element to the destination would not violate a predefined rule,

modifying the corresponding code of the source to reflect the removal of the link between the source and the destination; and

adding new code to the corresponding code of the other element to reflect the new link between the other element and the destination element.

109. (original): The computer-readable medium of claim 108, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is a class and whether the destination is

another class; and

when it is determined that the other element is the class and that the destination is the other class,

identifying the new link between the other element and the destination as an inheritance link.

110. (original): The computer-readable medium of claim 108, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is a class and whether the destination is an interface; and

when it is determined that the other element is the class and that the destination is the interface,

identifying the new link between the other element and the destination as an implementation link.

111. (original): The computer-readable medium of claim 110, wherein the method further comprises the step of identifying a link error when it is determined that the other element is the class and that the destination is not the interface.

112. (original): The computer-readable medium of claim 108, wherein the step of determining whether linking the other element to the destination would violate a predefined rule, comprises the steps of:

determining whether the other element is an interface and the destination is another interface; and

when it is determined that the other element is the interface and the destination is the other interface,

identifying the new link between the other element and the destination as an inheritance link.

113. (original): The computer-readable medium of claim 112, wherein the method further comprises the step of identifying a link error when it is determined that the other element is not the interface.

114. (original): The computer-readable medium of claim 112, wherein the method further comprises the step of identifying a link error when it is determined that the destination is not the other interface.

115. (cancelled).

116. (cancelled).

117. (cancelled).

118. (cancelled).

119. (cancelled).

120. (cancelled).

121. (cancelled).

122. (cancelled).

123. (cancelled).

124. (currently amended): A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system including a software development project having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names along with a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code in the project;

displaying the graphical representation of the corresponding code of each of

the plurality of elements including the source and destination by using the names and display coordinates correlated by the transient meta model;

receiving an identification of the link;

receiving a selection of one of the linked elements;

receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

determining whether the selected element is the source; and

when it is determined that the selected element is the source,

modifying the corresponding code of the source independently of the graphical representation in order to reflect a new link between the source and the other element and

modifying the graphical representation of the corresponding code of the source to reflect the new link between the source and the other element.

125. (cancelled).

126. (previously presented): The computer-readable medium of claim 185, wherein the method further comprises the step of modifying the graphical representation of the code corresponding to the source to reflect the removal of the link to the destination.

127. (original): The computer-readable medium of claim 124, wherein the method further comprises the steps of:

when it is determined that the selected element is the source,

determining whether linking the source to the other element would violate a predefined rule; and

when it is determined that linking the source to the other element would not violate a predefined rule,

modifying the corresponding code of the source to reflect the removal of the link between the source and the destination; and

adding new code to the corresponding code of the source to reflect the new link to the other element.

128. (cancelled).

129. (original):The computer-readable medium of claim 127, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is a class and whether the other element is another class; and

when it is determined that the source is the class and that the other element is the other class,

identifying the new link between the source and the other element as an

inheritance link.

130. (original): The computer-readable medium of claim 127, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is a class and whether the other element is an interface; and

when it is determined that the source is the class and that the other element is the interface,

identifying the new link from the source to the other element as an implementation link.

131. (original): The computer-readable medium of claim 130, wherein the method further comprises the step of identifying a link error when it is determined that the other element is not the interface.

132. (original): The computer-readable medium of claim 127, wherein the step of determining whether linking the source to the other element would violate a predefined rule, comprises the steps of:

determining whether the source is an interface and the other element is another interface; and

when it is determined that the source is the interface and the other element is the other interface,

identifying the new link between the source and the other element as an inheritance link.

133. (original): The computer-readable medium of claim 132, wherein the method further comprises the step of identifying a link error when it is determined that the source is not the interface.

134. (original): The computer-readable medium of claim 132, wherein the method further comprises the step of identifying a link error when it is determined that the other element is not the other interface.

135. (cancelled).

136. (cancelled).

137. (cancelled).

138. (cancelled).

139. (cancelled).

140. (cancelled).

141. (cancelled.

142. (cancelled).

143. (cancelled).

144. (cancelled).

145. (cancelled).

146. (cancelled).

147. (cancelled).

148. (cancelled).

149. (cancelled).

150. (currently amended): A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system including a software development project having a plurality of elements having a graphical representation stored in a graphical view file that includes symbols having

display coordinates and a corresponding code stored in a source code file that includes names, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including a first and second element by using the using the names and display coordinates correlated by the transient meta model;

receiving an identification of a first of the plurality of elements;

receiving an identification of a second of the plurality of elements;

receiving an indication that the first element is to be included in the second element;

determining whether the inclusion of the first element in the second element would violate a predefined rule; and

when it is determined that the inclusion of the first element in the second element would not violate a predefined rule,

transferring the code corresponding to the first element into the second element, wherein the code transfer occurs independently of the graphical representation and

modifying the graphical representation of the code of the second element

to reflect the transfer of the code corresponding to the first element into the second element.

151. (cancelled).

152. (cancelled).

153. (previously presented): The computer-readable medium of claim 150, wherein the step of transferring code comprises the steps of:

removing the code corresponding to the first element from a file corresponding to the second element;

placing the code corresponding to the first element within the code corresponding to the second element; and

deleting the file corresponding to the first element.

154. (original): The computer-readable medium of claim 150, wherein the method further comprises the steps of:

when it is determined that the first element is the class and that the second element is not the other class,

determining whether the second element is a package; and

when it is determined that the second element is a package,

moving a file that includes code corresponding to the first element to a directory associated with the second element.

155. (cancelled).

156. (cancelled).

157. (currently amended): A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system including a software development project having a plurality of elements, each element having corresponding code stored in source code file that includes names and a graphical representation stored in a graphical view file that includes symbols and display coordinates, wherein code corresponding to a first of the plurality of elements is nested in the code corresponding to a second of the plurality of elements, the method comprising the steps of:

generating a transient meta model that stores a language neutral representation of the source code;

reading the graphical view file into the transient meta model to correlate the names and display coordinates of each symbol of the corresponding source code file in the project;

displaying the graphical representation of the corresponding code of each of the plurality of elements including the first element and the second element by using the names and display coordinates correlated by the transient meta model;

receiving an indication that the first element is to be removed from the second

element;

determining whether the removal of the first element from the second element would violate a predefined rule; and

when it is determined that the removal of the first element from the second element would not violate a predefined rule,

removing the code corresponding to the first element from the second element, wherein the code removal occurs independently of the graphical representation and

modifying a graphical representation of the code corresponding to the second element to reflect the removal of the first element from the second element.

158. (original): The computer-readable medium of claim 157, wherein the method further comprises the step of placing the code corresponding to the first element into a file.

159. (cancelled).

160. (cancelled).

161. (previously presented): The computer-readable medium of claim 157, wherein the method further comprises the steps of:

when it is determined that the first element is the class and that the second element is not another class,

determining whether the second element is a package; and

when it is determined that the second element is a package,

removing a first file that includes code corresponding to the first element from a directory associated with the second element and placing the first file in another directory.

162. (cancelled).

163. (cancelled).

164. (cancelled).

165. (currently amended): A data processing system for developing a software project comprising:

a secondary storage device further comprising a plurality of elements, each element having corresponding code stored in a source file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates;

a memory device further comprising a program that generates a transient meta model that correlates the names and display coordinates of each symbol of the corresponding source code file in the project,

that displays the graphical representation of the corresponding code of each of the plurality of elements including a first element and a second element, by using the symbols and display coordinates correlated by the transient meta model,

that receives a request to form a link,

that receives an indication of a first of the plurality of elements,

that receives an indication of a second of the plurality of elements,

that determines whether linking the first element to the second element would violate a predefined rule,

that generates new code independent of the graphical representation and adds the new code to the first element to reflect the link to the second element when it is determined that linking the first element to the second element would not violate a predefined rule, and

that modifies the graphical representation of the code associated with the first element to reflect the link to the second element; and

a processor for running the program.

166. (cancelled).

167. (cancelled).

168. (original): The data processing system of claim 165, wherein when the program determines whether linking the first element to the second element would violate a predefined rule, the program determines whether the first element is a class and whether the second element is another class, and when it is determined that the first element is the class and that the second element is the other class, the program identifies the link from the first element to the second element as an inheritance link.

169. (original): The data processing system of claim 165, wherein when the program determines whether linking the first element to the second element would violate a predefined rule, the program determines whether the first element is a class and whether the second element is an interface, and when it is determined that the first element is the class and that the second element is the interface, the program identifies the link from the first element to the second element as an implementation link.

170. (original): The data processing system of claim 165, wherein when the program determines whether linking the first element to second element would violate a predefined rule, the program determines whether the first element is an interface and the second element is another interface, and when it is determined that the first element is the interface and the second element is the other interface, the program identifies the link from the first element to the second element as an inheritance link.

171. (currently amended): A data processing system <u>for developing a software project</u> comprising:

a secondary storage device further comprising a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names with a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination;

a memory device further comprising a program that generates a transient meta model that correlates the symbols associated with the display coordinates,

that displays the graphical representation of the corresponding code of each of the plurality of elements including the source and the destination by using the symbols and display coordinates correlated by the transient meta model,

that receives a selection of one of the linked elements,

that receives an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed,

that determines whether the selected element is the destination, and

that when it is determined that the selected element is the destination, generates new code independently of the graphical representation and adds the new code to the code corresponding to the other element to reflect the new link between the other

element and the destination when it is determined that the selected element is the destination,

removes a portion of the corresponding code of the source that reflects the link between the source and the destination,

modifies the graphical representation of the corresponding code of the source to reflect the removal of the link to the destination, and

modifies the graphical representation of the corresponding code of the other element to reflect the new link; and

a processor for running the program.

172. (original): The data processing system of claim 171, wherein when it is determined that the other element is the class and that the destination is not the other class, the program further determines whether the destination is an interface, and when it is determined that the destination is the interface, the program identifies the new link between the other element and the destination as an implementation link.

173. (original): The data processing system of claim 171, wherein when it is determined that the other element is not the class and that the destination is not the other class, the program further determines whether the other element is an interface and whether the destination is another interface, and when it is determined that the other element is the interface and that the destination is the other interface, the program identifies the new link between the other element and the destination as an inheritance link.

174. (cancelled).

175. (cancelled).

176. (previously presented): The data processing system of claim 187, wherein when it is determined that the source is the class and that the other element is not the other class, the program further determines whether the other element is in an interface, and when it is determined that the other element is the interface, the program identifies the new link between the source and the other element as an implementation link.

177. (previously presented): The data processing system of claim 187, wherein when it is determined that the source is not the class and that the other element is not the other class, the program further determines whether the source is an interface and the other element is another interface, and when it is determined that the source is the interface and the other element is the other interface, the program identifies the new link between the source and the other element as an inheritance link.

178. (currently amended): A data processing system for developing a software project comprising:

a secondary storage device further comprising a plurality of elements, each element having corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that

includes symbols having display coordinates; a memory device further comprising a program that generates a transient meta model that correlates the symbols associated with the display coordinates, that displays the graphical representation of the code of a first of the plurality of elements and a graphical representation of the code of a second of the plurality of elements by using the symbols and display coordinates correlated by the transient meta model, that receives an indication that the first element is to be included in the second element, that determines whether inclusion of the first element in the second element would violate a predefined rule,

that transfers code corresponding to the first element into the second element when it is determined that the inclusion of the first element in the second element would not violate a predefined rule, wherein the code transfer occurs independently of the graphical representation, and

that modifies a graphical representation of the code of the second element to reflect the transfer of the first element into the second element; and

a processor for running the program.

179. (previously presented): The data processing system of claim 178, wherein the program removes the code corresponding to the first element from a file, places the code corresponding to the first element within the code corresponding to the second element, and deletes the file corresponding to the first element.

180. (currently amended): A data processing system for developing a software project comprising:

a secondary storage device further comprising a plurality of elements having corresponding code stored in a source code file that includes names and graphical representations stored in a graphical view file that includes symbols having display coordinates, wherein a first of the plurality of elements is nested within a second of the plurality of elements; a memory device further comprising a program that generates a transient meta model that correlates the symbols associated with the display coordinates, that displays the graphical representation of the code of a first of the plurality of elements and a graphical representation of the code of a second of the plurality of elements by using the symbols and display coordinates correlated by the transient meta model, that receives an indication that the first element is to be removed from the second element, that determines whether the removal of the first element from the second element would violate a predefined rule, and

that removes the code corresponding to the first element from the second element when it is determined that the removal of the first element from the second element would not violate a predefined rule, wherein the code removal occurs independently of each graphical representation,

that modifies the graphical representation of the second element to reflect the removal of the first element from the second element, and

that places the code corresponding to the first element into a file; and

a processor for running the program.

181. (cancelled).

182. (previously presented): The data processing system of claim 180, wherein the first element is a class and the second element is not another class, the program further determines whether the second element is a package, and when it is determined that the second element is a package, the program removes the first file corresponding to the first element from a directory associated with the second element.

183. (currently amended): A system including a software development project having a plurality of elements, each element having corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates, the system comprising:

a memory device further comprising a program that generates a transient meta model for correlating the names and display coordinates of each symbol of the corresponding source code file in the project;

~~means for~~ a computer monitor for displaying the graphical representation of the corresponding code of each of the plurality of elements by using the names and display coordinates correlated by the transient meta model;

means for receiving a request to form a link;

means for receiving an indication of a first of the plurality of elements;

means for receiving an indication of a second of the plurality of elements; and

means for generating new code independent of the graphical representation and adding the new code to the first element to reflect the link to the second element in response to receiving the request, the indication of the first element, and the indication of the second element; and

means for modifying the graphical representation of the code associated with the first element to reflect the link to the second element.

184. (previously presented): The method of claim 42, further comprising the step of modifying the code corresponding to the source to reflect the removal of the link to the destination.

185. (previously presented): The computer-readable medium of claim 124, wherein the method further comprises the step of modifying the code corresponding to the source to reflect the removal of the link to the destination.

186. (previously presented): The data processing system of claim 171, wherein when it is determined that the other element is a class and that the destination is another class, the program identifies the new link between the other element and the destination as an inheritance link.

187. (currently amended): A data processing system for developing a software project comprising:

a secondary storage device further comprising a plurality of elements and having

a link between two of the plurality of elements, wherein each element has corresponding code stored in a source file that includes names and a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination;

a memory device further comprising a program that generates a transient meta model that correlates the symbols that correspond to the display coordinates,

that displays the graphical representation of the corresponding code of each of the plurality of elements including the source and the destination by using the symbols and display coordinates correlated by the transient meta model,

that receives a selection of one of the linked elements,

that receives an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed,

that determines whether the selected element is the source,

that when the element is the source,

removes a portion of the corresponding code of the source that reflects the link between the source and the destination,

generates new code independently of the graphical representation and adds the, new code to the code corresponding to the

source to reflect the new link between the source and the other element,

modifies the graphical representation of the corresponding code of the source to reflect the removal of the link to the destination, and

modifies the graphical representation of the corresponding code of the source to reflect the new link to the other element; and

a processor for running the program.

188. (previously presented): The data processing system of claim 187, wherein when it is determined that the source is a class and that the other element is another class, the program identifies the new link between the source and the other element as an inheritance link.

189. (previously presented): The data processing system of claim 180, wherein the method further comprises the step of placing the code corresponding to the first element into a file.

190. (currently amended) A system including a software development project having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names and a graphical representation stored in a graphical view file that includes symbols

having display coordinates and the linked elements include a source and a destination, the system comprising:

a memory device further comprising a program that generates a transient meta model for correlating the names and display coordinates of each symbol of the corresponding source code file project;

~~means for~~ a computer monitor for displaying, the graphical representation of the corresponding code of each of the plurality of elements including the source and the destination by using the names and display coordinates correlated by the transient meta model;

means for receiving an identification of the link;

means for receiving a selection of one of the linked elements;

means for receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

means for determining whether the selected element is the destination; and

means for :

modifying the corresponding code of the other element to reflect a new link between the other element and the destination elements, wherein the code modification occurs independently of the graphical representation, and

modifying the graphical representation of the corresponding code of the other element to reflect the new link between the other element and the destination element

when it is determined that the selected element is the destination.

191. (currently amended): A system including a software development project having a plurality of elements and having a link between two of the plurality of elements, wherein each element has corresponding code stored in a source code file that includes names having a graphical representation stored in a graphical view file that includes symbols having display coordinates and the linked elements include a source and a destination, the system comprising:

a memory device further comprising a program that generates a transient meta model for correlating the names and display coordinates of each symbol of the corresponding source code file in the project;

~~means for~~ a computer monitor for displaying a computer monitor for, the graphical representation of the corresponding code of each of the plurality of elements including the source and destination by using the names and display coordinates correlated by the transient meta model;

means for receiving an identification of the link;

means for receiving a selection of one of the linked elements;

means for receiving an identification of another of the plurality of elements that is different than the linked elements, wherein a graphical representation of the corresponding code of the other element is displayed;

means for:

modifying the corresponding code of the source to reflect a new link

between the source and the other element wherein the corresponding code modification occurs independently of the graphical representation, and

modifying the graphical representation of the corresponding code of the source to reflect the new link between the source and the other element,

when it is determined that the selected element is the source.

192. (currently amended) A system including a software development project having a plurality of elements, each having a graphical representation the system comprising:

a memory device further comprising a program that generates a transient meta model for correlating the names and display coordinates of each symbol of the corresponding source code file in the project;

~~means for~~ a computer monitor for displaying, the graphical representation of the corresponding code of each of the plurality of elements including a first and a second element by using the names and display coordinates correlated by the transient meta model;

means for receiving an identification of a first of a plurality of elements;

means for receiving an identification of a second of the plurality of elements;

means for receiving an indication that the first element is to be included in the second element;

means for transferring code corresponding to the first into the second element,

wherein the code transfer occurs independently of the graphical representation; and

means for modifying the graphical representation of the code of the second element to reflect the transfer of the code corresponding to the first element into the second element.

193. (currently amended): A system including a software development project having a plurality of elements, each element having corresponding code and a graphical representation, wherein code corresponding to a first of the plurality of elements is nested in code corresponding to a second of the plurality of elements, the system comprising:

a memory device further comprising a program that generates a transient meta model for correlating the names and display coordinates of each symbol of the corresponding source code file in the project;

~~means for~~ a computer monitor for displaying, the graphical representation of the corresponding code of each of the plurality of elements including the first element and the second element by using the names and display coordinates correlated by the transient meta model;

means for receiving an indication that the first element is removed from the second element;

means for removing the code corresponding to the first element from the second element, wherein said code removal occurs independently of the graphical representation; and

means for modifying a graphical representation of the code corresponding to the

second element to reflect the removal of the first element from the second element.

194. (previously presented): The data processing system in claim 165, further comprising a language-neutral representation of the source code, wherein the language neutral representation of the source code is used to generate a graphical representation of the source code and a textual representation of the source code.

195. (previously presented): The data processing system in claim 194, wherein modifications to he textural representation of the source code will immediately modify the corresponding graphical representation of the source code and vice versa, thereby allowing simultaneous viewing of the textual and graphical representations of the source code.

196. (previously presented): The data processing system in claim 194, wherein the language neutral representation of the source code is generated from a multiplicity of programming languages.

197. (previously presented): The data processing system in claim 165, wherein the source code is used directly to generate a graphical representation of the source code and a textual representation of the source code.

198. (previously presented): The data processing system in claim 171, further comprising a language-neutral representation of the source code, wherein the language neutral representation of the source code is used to generate a graphical representation of the source code and a textual representation of the source code.

199. (previously presented): The data processing in claim 197, wherein modifications to the textual representation of the source code will immediately modify the corresponding graphical representation of the source code and vice versa, thereby allowing simultaneous viewing of the textual and graphical representation of the source code.

200. (previously presented) The data processing system in claim 197, wherein the language neutral representation of the source code is generated from a multiplicity of programming languages.

201. (previously presented): The data processing system in claim 171, wherein the source code is used directly to generate a graphical representation of the source code and a textual representation of the source code.

202. (previously presented): The data processing system of claim 187, further comprising a language-neutral representation of the source code, wherein the language neutral representation of the source code is used to generate a graphical representation of the source code and a textual representation of the source code.

203. (previously presented): The data processing system in claim 202, wherein modifications to the textual representation of the source code will immediately modify the corresponding graphical representation of the source code and vice versa, thereby allowing simultaneous viewing of the textual and graphical representations of the source code.

204. (previously presented): The data processing system in claim 202, wherein the language neutral representation of the source code is generated from a multiplicity of programming languages.

205. (previously presented): The data processing in claim 187, wherein the source code is used directly to generate a graphical representation of the source code and a textual representation of the source code.

206. (previously presented): The data processing system in claim 178, further comprising a language-neutral representation of the source code, wherein the language neutral representation of the source code is used to generate a graphical representation of the source code and a textual representation of the source code.

207. (previously presented): The data processing system in claim 206, wherein modifications to the textual representation of the source code will immediately modify the corresponding graphical representation of the source code and vice versa, thereby

allowing simultaneous viewing of the textual and graphical representations of the source code.

208. (previously presented): The data processing system in claim 206, wherein the language neutral representation of the source code is generated from a multiplicity of programming languages.

209. (previously presented): The data processing system in claim 178, wherein the source code is used directly to generate a graphical representation of the source code and a textual representation of the source code.

210. (previously presented): The data processing system of claim 180, further comprising a language-neutral representation of the source code, wherein the language neutral representation of the source code is used to generate a graphical representation of the source code and a textual representation of the source code.

211. (previously presented): The data processing system of claim 210, wherein modifications to the textual representation of the source code will immediately modify the corresponding graphical representation of the source code and vice versa, thereby allowing simultaneous viewing of the textual and graphical representations of the source code.

212. (previously presented): The data processing system in claim 210, wherein the language neutral representation of the source code is generated from a multiplicity of programming languages.

213. (previously presented): The data processing system in claim 180, wherein the source code is used directly to generate a graphical representation of the source code and a textual representation of the source code.